0828.64986 PATENT APPLICATION

#2
cpc
3/22

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In Re U.S. Patent Application)
)
Applicant: Takahara et al.)
)
Serial No.)
)
Filed: December 4, 2000)
)
For: INFORMATION PROCESSING)
APPARATUS)
)
Art Unit:)

I hereby certify that this paper is being deposited with the United States Postal Service as EXPRESS mail in an envelope addressed to: Assistant Commissioner for Patents, Washington, D.C. 20231, on December 4, 2000.
Express Label No.: EL769180898US
Signature: _____

CLAIM FOR PRIORITY

Assistant Commissioner for Patents
Washington, DC 20231

Sir:

Applicants claim foreign priority benefits under 35 U.S.C. § 119 on the basis of the foreign application identified below:

Japanese Patent Application No. 2000-072446, filed March 10, 2000.

A certified copy of the priority document is enclosed.

Respectfully submitted,

GREER, BURNS & CRAIN, LTD.

By: _____
James K. Folker
Reg. No. 37,538

December 4, 2000
300 South Wacker Drive
Suite 2500
Chicago, IL 60606
(312) 360-0080

0828.64986
(312) 360 0080

日 本 国 特 許 庁
PATENT OFFICE
JAPANESE GOVERNMENT

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。
This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日
Date of Application: ２０００年 ３月１０日

出 願 番 号
Application Number: 特願２０００－０７２４４６

出 願 人
Applicant (s): 富士通株式会社

２０００年 ８月２５日

特許庁長官
Commissioner,
Patent Office 及川耕造

出証番号 出証特２０００－３０６８２０２

【書類名】　特許願
【整理番号】　0050005
【提出日】　平成12年 3月10日
【あて先】　特許庁長官殿
【国際特許分類】　G06F 09/45
【発明の名称】　情報処理装置
【請求項の数】　12
【発明者】
【住所又は居所】　神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内
【氏名】　高原　浩二
【発明者】
【住所又は居所】　神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内
【氏名】　青木　正樹
【特許出願人】
【識別番号】　000005223
【氏名又は名称】　富士通株式会社
【代理人】
【識別番号】　100092152
【弁理士】
【氏名又は名称】　服部　毅巖
【電話番号】　0426-45-6644
【手数料の表示】
【予納台帳番号】　009874
【納付金額】　21,000円
【提出物件の目録】
【物件名】　明細書　1
【物件名】　図面　1

【物件名】　　要約書　1

【包括委任状番号】　9705176

【プルーフの要否】　　要

【書類名】　明細書

【発明の名称】　情報処理装置

【特許請求の範囲】

【請求項１】　動的変数を含むソースファイルをコンパイル処理によってオブジェクトファイルに翻訳し、リンク処理によって実行可能形式のロードモジュールに変換する情報処理装置において、

前記ソースファイルから対象となる動的変数を特定する動的変数特定手段と、

前記動的変数特定手段によって特定された動的変数が、前記ロードモジュールの実行時にメモリ上に展開される際に確保される領域を特定する領域特定手段と、

前記領域特定手段によって特定された領域を、所定の初期値によって初期化する初期化手段と、

を有することを特徴とする情報処理装置。

【請求項２】　前記動的変数特定手段によって複数の異なる動的変数が特定された場合には、前記メモリ上に展開される際に確保される領域を必要に応じて統合する領域統合手段を更に有することを特徴とする請求項１記載の情報処理装置。

【請求項３】　前記リンク処理によって統合される１または２以上のオブジェクトファイル内に同一の動的変数が散在している場合には、これらを統合する変数統合手段を更に有することを特徴とする請求項１記載の情報処理装置。

【請求項４】　前記動的変数特定手段は、対象となる動的変数を新たなデータセクションに割り当て、

前記初期化手段は、前記新たなデータセクションに割り当てられた動的変数のみを対象として初期化処理を実行することを特徴とする請求項１記載の情報処理装置。

【請求項５】　前記初期化手段は、複数のオブジェクトファイルをリンクして前記ロードモジュールを生成する際に、所定のオブジェクトファイルでは前記新たなデータセッションに割り当てられ、他のオブジェクトファイルでは通常のデータセクションに割り当てられている動的変数が存在する場合には、その動的

変数に関しては初期化処理を実行しないことを特徴とする請求項４記載の情報処理装置。

【請求項６】　前記初期化手段による初期値を、コンパイル処理前に入力する初期値入力手段を更に有することを特徴とする請求項１記載の情報処理装置。

【請求項７】　前記初期化手段による初期値を、前記ロードモジュールの実行前に入力する初期値入力手段を更に有することを特徴とする請求項１記載の情報処理装置。

【請求項８】　前記初期化の対象となる動的変数を指定する初期化変数指定手段を更に有することを特徴とする請求項１記載の情報処理装置。

【請求項９】　前記ロードモジュールの実行時において、前記初期値が設定されたままの動的変数が参照された場合には、エラーを通知するエラー通知手段を更に有することを特徴とする請求項１記載の情報処理装置。

【請求項１０】　前記動的変数が配列である場合には、ソースコードにおいて宣言された配列の要素数よりも所定量だけ多いメモリ領域を確保するメモリ確保手段を更に有し、

前記初期化手段は、前記メモリ確保手段によって確保されたメモリ上の全ての領域を所定の初期値によって初期化し、

前記エラー通知手段は、前記初期値が設定されたままの配列が参照された場合には、エラーを通知することを特徴とする請求項９記載の情報処理装置。

【請求項１１】　動的変数を含むソースファイルをコンパイル処理によってオブジェクトファイルに翻訳し、リンク処理によって実行可能形式のロードモジュールに変換する処理をコンピュータに実行させるプログラムを記録したコンピュータ読み取り可能な記録媒体において、

コンピュータを、

前記ソースファイルから対象となる動的変数を特定する動的変数特定手段、

前記動的変数特定手段によって特定された動的変数が、前記ロードモジュールの実行時にメモリ上に展開される際に確保される領域を特定する領域特定手段、

前記領域特定手段によって特定された領域を、所定の初期値によって初期化する初期化手段、

として機能させるプログラムを記録したコンピュータ読み取り可能な記録媒体。

【請求項１２】　配列を含むソースファイルをコンパイル処理によってオブジェクトファイルに翻訳し、リンク処理によって実行可能形式のロードモジュールに変換する情報処理装置において、

前記ソースファイルから対象となる配列を特定する配列特定手段と、

前記配列特定手段によって特定された配列において宣言されている領域よりも所定のバイト数だけ多い領域を、前記ロードモジュールの実行時においてメモリ上に確保する領域確保手段と、

前記領域確保手段によって確保された領域を所定の初期値で初期化する初期化手段と、

を有することを特徴とする情報処理装置。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は情報処理装置に関し、特に、動的変数を含むソースコードをコンパイル処理によりオブジェクトコードに翻訳し、リンク処理により実行可能形式のロードモジュールに変換して実行する情報処理装置に関する。

【０００２】

【従来の技術】

ＦＯＲＴＲＡＮやＣ言語のような言語で使用されている変数には、実行時のメモリへの配置の態様により、静的変数と動的変数の２種類に分類することができる。静的変数は、プログラムの実行過程によらず記憶領域への割り当てが不変である変数であり、また、動的変数は、プログラムの実行中に記憶領域の割り当てが行われる変数である。

【０００３】

ところで、これらの言語によってプログラムを記述する場合、使用する変数に対しては初期値を設定する必要がある。初期値を設定しないで変数を参照した場合には、変数の値が未定であるため、予期せぬエラーが発生する場合が想定され

るからである。

【０００４】

同様の不具合は配列の場合にも存在する。即ち、配列の場合では、配列添字が宣言された範囲を超えた場合には未定義の領域を参照することになるので、その場合にも前述の場合と同様の問題が発生することになる。

【０００５】

【発明が解決しようとする課題】

従来において、初期設定されていない変数が参照されていることの検査（以下、未定義参照の検査と称する）としては、次の２つの方法が一般的に用いられていた。

（１）対象となる変数に所定の初期値を設定する命令をプログラムに追加し、プログラムの実行時において、その所定の値が検出された場合には、未定義の変数が参照されていると判断する。

（２）コンパイル時に、対象となる変数に所定の初期値をコンパイラが代入し、プログラムの実行時にその値が検出された場合には、未定義の変数が使用されていると判断する。

【０００６】

また、宣言されていない範囲の配列添字が参照されていることの検査（以下、配列添字の検査と称する）としては、次の２つの方法が一般的に用いられていた。

（３）配列添字をプリントアウトする命令をプログラムに追加するなどして、配列添字の内容をプログラマが直接参照することにより判断する。

（４）コンパイル時に、配列が参照されている命令をプログラムから検出し、その命令の前後に配列添字が適正な範囲に存在するか否かをチェックする新たな命令をコンパイラが追加する。

【０００７】

ところで、（１），（３）の方法の場合では、新たな命令をプログラマが手入力で追加する必要があるため、煩雑であるという問題点があった。

また、（２），（４）の方法の場合、プログラムを再度コンパイルする必要が

生じ、コンパイルに長時間を要するプログラムに対しては効率がよくないという問題点があった。

【０００８】

更に、（２），（４）の方法では、前述の動的変数に対しては、初期値を設定することができないという問題点があった。この点について以下に詳述する。

即ち、動的変数は、プログラムが実行される際にメモリ領域への割り当てが初めて決定される変数であり、その割り当て先はＯＳ（Operating System）が管理している。従って、変数の内容を所定の値で初期化するためには、ＯＳが割り当てた領域を知る必要があるが、そのような方法は従来存在していなかったため、動的変数に関しては任意の値への初期化が困難であった。

【０００９】

本発明は、このような点に鑑みてなされたものであり、デバッグオプションにより動的変数を任意の値で初期化することが可能な情報処理装置を提供することを目的とする。

【００１０】

【課題を解決するための手段】

本発明では上記課題を解決するために、図１に示す、動的変数を含むソースファイル１をコンパイル処理によってオブジェクトファイルに翻訳し、リンク処理によって実行可能形式のロードモジュールに変換する情報処理装置において、前記ソースファイル１から対象となる動的変数を特定する動的変数特定手段２と、前記動的変数特定手段２によって特定された動的変数が、前記ロードモジュールの実行時にメモリ５上に展開される際に確保される領域を特定する領域特定手段３と、前記領域特定手段３によって特定された領域を、所定の初期値によって初期化する初期化手段４と、を有することを特徴とする情報処理装置が提供される。　動的変数特定手段２は、ソースファイル１から対象となる動的変数を特定する。領域特定手段３は、動的変数特定手段２によって特定された動的変数が、ロードモジュールの実行時にメモリ５上に展開される際に確保される領域を特定する。初期化手段４は、領域特定手段３によって特定された領域を、所定の初期値によって初期化する。

【００１１】

また、配列を含むソースファイルをコンパイル処理によってオブジェクトファイルに翻訳し、リンク処理によって実行可能形式のロードモジュールに変換する情報処理装置において、前記ソースファイルから対象となる配列を特定する配列特定手段と、前記配列特定手段によって特定された配列において宣言されている領域よりも所定のバイト数だけ多い領域を、前記ロードモジュールの実行時においてメモリ上に確保する領域確保手段と、前記領域確保手段によって確保された領域を所定の初期値で初期化する初期化手段と、を有することを特徴とする情報処理装置が提供される。

【００１２】

ここで、配列特定手段は、ソースファイルから対象となる配列を特定する。領域確保手段は、配列特定手段によって特定された配列において宣言されている領域よりも所定のバイト数だけ多い領域を、ロードモジュールの実行時においてメモリ上に確保する。初期化手段は、領域確保手段によって確保された領域を所定の初期値で初期化する。

【００１３】

【発明の実施の形態】

以下、本発明の実施の形態を図面を参照して説明する。

図１は、本発明の動作原理を説明する原理図である。この図において、ソースコードは、例えば、ＦＯＲＴＲＡＮやＣ言語で記述されたプログラムであり、動的変数を含んでいる。

【００１４】

動的変数特定手段２は、ソースファイル１から対象となる動的変数を特定する。

領域特定手段３は、動的変数特定手段２によって特定された動的変数が、実行時においてメモリ上に展開される際に確保される領域を特定する。

【００１５】

初期化手段４は、領域特定手段３によって特定された領域を、所定の初期値によって初期化する。

メモリ５には、ソースファイル１がコンパイルされ、リンクされて生成されたロードモジュールが格納される。

【００１６】

次に、以上の原理図の動作について説明する。

いま、動的変数を含むソースファイル１が入力されたとすると、動的変数特定手段２は、ソースファイル１から動的変数を特定する。

【００１７】

領域特定手段３は、動的変数特定手段２によって特定された動的変数がメモリ５に展開された際の領域を特定する。具体的には、領域特定手段３は、オブジェクトファイルにおいて、対象となる動的変数に係るオブジェクトコードが格納されている相対アドレス（先頭アドレスと末尾アドレス）を取得し、初期化手段４に対して供給する。

【００１８】

初期化手段４は、プログラムが実行された際に、そのオブジェクトコードが格納されたメモリの先頭アドレス（絶対アドレス）と、前述の先頭アドレスおよび末尾アドレス（相対アドレス）とから、動的変数が格納されるメモリの絶対アドレスを特定し、デバッグオプション等によって指定された初期値（この例では“８Ｂ”）によってその領域を初期化する。

【００１９】

その結果、メモリ５の動的変数が割り当てられた領域は、所定の初期値によって初期化されることになる。

以上に説明したように、本発明に係る情報処理装置によれば、動的変数を任意の値で初期化することが可能となるので、動的変数についてもデバッグオプションの指定により未定義参照の検査が可能となる。

【００２０】

次に、図２および図３を参照して、本発明の実施の形態について説明する。

図２は、本発明の実施の形態の構成例を示す図である。この図に示すように、本発明に係る情報処理装置１０は、ＣＰＵ（Central Processing Unit）１０ａ、ＲＯＭ（Read Only Memory）１０ｂ、ＲＡＭ（Random Access Memory）１０ｃ

、ＨＤＤ（Hard Disk Drive）１０ｄ、ＧＣ（Graphics Card）１０ｅ、Ｉ／Ｆ（Interface）１０ｆ、および、バス１０ｇによって構成されており、外部には表示装置１１および入力装置１２が接続されている。

【００２１】

ここで、ＣＰＵ１０ａは、装置の各部を制御するとともに、ＨＤＤ１０ｄに格納されているプログラム等に応じて各種演算処理を実行する。

ＲＯＭ１０ｂは、ＣＰＵ１０ａが実行する基本的なプログラムやデータ等を格納している。

【００２２】

ＲＡＭ１０ｃは、ＣＰＵ１０ａが実行途中のプログラムや演算途中のデータ等を一時的に格納する。

ＨＤＤ１０ｄは、システム全体を制御するＯＳ、コンパイラ、デバッガ、リンカ、および、コンパイルの対象となるソースファイルやコンパイル後のオブジェクトファイル等を格納している。

【００２３】

ＧＣ１０ｅは、ＣＰＵ１０ａから供給された描画命令に従って描画処理を実行し、得られた画像データを映像信号に変換して表示装置１１に出力する。

Ｉ／Ｆ１０ｆは、入力装置１２から供給されたデータの表現形式を変換して、入力する。

【００２４】

バス１０ｇは、ＣＰＵ１０ａ、ＲＯＭ１０ｂ、ＲＡＭ１０ｃ、ＨＤＤ１０ｄ、ＧＣ１０ｅ、および、Ｉ／Ｆ１０ｆを相互に接続し、これらの間でデータの授受を可能とする。

【００２５】

表示装置１１は、例えば、ＣＲＴ（Cathode Ray Tube）モニタによって構成され、ＧＣ１０ｅから供給された画像信号を表示出力する。

入力装置１２は、例えば、キーボードまたはマウスによって構成され、ユーザの操作に応じたデータを発生して、Ｉ／Ｆ１０ｆに供給する。

【００２６】

ここで、ＯＳ、コンパイラ、リンカ、ソースファイル、および、オブジェクトファイルは、ＨＤＤ１０ｄに格納されており、必要に応じて読み出されてＲＡＭ１０ｃに展開され、ＣＰＵ１０ａによって実行される。

【００２７】

図３は、ＯＳ２０、コンパイラ２０ａ、リンカ２０ｂ、ソースファイル２１、ライブラリ２２、および、ロードモジュール２３の対応関係を模式的に示した図である。

【００２８】

この図に示すように、ＯＳ２０は、コンパイラ２０ａおよびリンカ２０ｂの実行を管理するとともに、これらからの要求に応じてＨＤＤ１０ｄに格納されているソースファイル２１およびライブラリ２２を読み出すとともに、生成されたロードモジュール２３をＨＤＤ１０ｄに対して格納する。また、ロードモジュールに対する実行要求が入力装置１２からなされた場合には、ＨＤＤ１０ｄに格納されている該当するロードモジュールを読み出して実行する。

【００２９】

ここで、ライブラリ２２は、例えば、数学の関数を演算するための数学関数ライブラリや、ロードモジュールを実行する際の初期化処理を行う初期化ライブラリ等によって構成されており、リンカ２０ｂによってリンク処理を実行する際に、必要に応じて読み出される。

【００３０】

次に、図４を参照して、本発明の実施の形態の動作の概要について説明する。先ず、コンパイラ２０ａに対して、ソースファイル３０が供給されると、コンパイラ２０ａは、ソースファイル３０に含まれている動的変数を検出する。この例では、「ＤＩＭＥＮＳＩＯＮ　Ａ（１０００）」によって宣言されている配列Ａ（初期値無しのローカル変数）と、「ＣＯＭＭＯＮ　／ＢＬＫ／Ｂ（１０００）」によって宣言されている配列Ｂ（初期値無しのグローバル変数）とが目的とする動的変数であるとして検出される。

【００３１】

次に、コンパイラ２０ａは、検出した動的変数を、新たなデータセクション（

以下、新データセクションと称する）の変数としてオブジェクトファイルに埋め込み、ソースファイル３０に記述された他の命令群に対するコンパイル処理を施した後、得られたオブジェクトファイル３１を出力する。

【００３２】

この例では、オブジェクトファイル３１には、新データセクションを示す「ｎｅｗ１」が付与されたオブジェクトコード「．ｂｓｓｎｅｗ１　ｍａｉｎ．ｌｏｃａｌ，４０００」と「．ｃｏｍｍｏｎｎｅｗ１　ｂｌｋ，４０００」とが記載されている。ここで、「ｂｓｓ」はセクション名を、「ｍａｉｎ」および「ｌｏｃａｌ」はメイン関数に属するローカル変数であることを、また、「４０００」は４０００バイト（単精度実数は４バイトから構成されるので）の領域を確保することを示す。また、「ｃｏｍｍｏｎ」はコモン変数であることを、また、「ｂｌｋ」は確保する領域に対して付与された名称である。

【００３３】

なお、従来のコンパイラで同様のプログラムをコンパイル処理した場合には、「．ｂｓｓ　ｍａｉｎ．ｌｏｃａｌ，４０００」と「．ｃｏｍｍｏｎ　ｂｌｋ，４０００」とが出力され、実質的な差異は「ｎｅｗ１」のみである。

【００３４】

リンカ２０ｂは、コンパイラ２０ａから出力されたオブジェクトファイル３１に対してリンク処理を施す。即ち、リンカ２０ｂは、先ず、ライブラリ２２から初期化ライブラリ２２ａおよび必要なライブラリを読み出し、オブジェクトファイル３１に対して付加する。

【００３５】

ここで、初期化ライブラリ２２ａは、ロードモジュールを実行した場合に、最初に実行すべき命令群によって構成されており、本実施の形態では、この初期化ライブラリ２２ａによって、目的とする動的変数の初期化処理を実行する。

【００３６】

続いて、リンカ２０ｂは、オブジェクトファイル３１に含まれている新データセクションの変数を検索する。新データセクションの変数が検出された場合には、その変数が格納されている領域の先頭の相対アドレス（オブジェクトプログラ

ムの先頭からの距離）であるｂｓｓ＿ｓｔａｒｔと、末尾の相対アドレスｂｓｓ＿ｅｎｄとを取得する。そして、取得した相対アドレスｂｓｓ＿ｓｔａｒｔと、ｂｓｓ＿ｅｎｄとを、先に付与した初期化ライブラリ２２ａに対して引き渡す（例えば、所定の位置に埋め込む）。

【００３７】

続いて、リンカ２０ｂは、リンク処理によってオブジェクトファイルを統合して実行可能形式のロードモジュール３２を生成する。

ロードモジュール３２が実行されると、ＯＳ２０は、ＨＤＤ１０ｄからロードモジュール３２を読み出し、ＲＡＭ１０ｃ上の所定の領域に配置する。

【００３８】

ＲＡＭ１０ｃ上にロードモジュール３２が配置されると、先ず、初期化ライブラリ２２ａが実行され、相対アドレスｂｓｓ＿ｓｔａｒｔ，ｂｓｓ＿ｅｎｄと、メモリ上に展開されたロードモジュールの先頭アドレス（絶対アドレス）とから、初期化すべき領域の先頭アドレスａｂｓｓ＿ｓｔａｒｔと末尾アドレスａｂｓｓ＿ｅｎｄ（絶対アドレス）とを算出する。そして、算出された先頭アドレスａｂｓｓ＿ｓｔａｒｔと末尾アドレスａｂｓｓ＿ｅｎｄによって指定されたメモリ３３の領域に対して、例えば、プログラムが実行される際に指定された初期値（この例では、“８Ｂ”）が書き込まれることになる。

【００３９】

以上の処理によれば、メモリ上に配置されるまでその格納位置が未定である動的変数に対しても、任意の値による初期化処理が可能となるので、デバッグオプションを利用して、未定義参照の検査を行うことが可能となる。

【００４０】

次に、以上に説明した処理を実現するフローチャートについて説明する。

図５は、コンパイラ２０ａが実行する処理の一例を説明するフローチャートである。このフローチャートが開始されると、以下の処理が実行される。

［Ｓ１］コンパイラ２０ａは、コンパイル処理の対象となるソースファイルから所定の変数を抽出する。

［Ｓ２］コンパイラ２０ａは、抽出した変数が動的変数であるか否かを判定し、

動的変数である場合にはステップＳ３に進み、それ以外の場合にはステップＳ４に進む。

［Ｓ３］コンパイラ２０ａは、新データセクションに係るオブジェクトコードを出力する。

［Ｓ４］コンパイラ２０ａは、通常データセクションに係るオブジェクトコードを出力する。

［Ｓ５］コンパイラ２０ａは、ソースファイルに未処理の変数が存在するか否かを判定し、存在する場合にはステップＳ１に戻って同様の処理を繰り返し、それ以外の場合にはステップＳ６に進む。

［Ｓ６］コンパイラ２０ａは、変数以外の部分に対するコンパイル処理を実行する。

【００４１】

図６は、リンカ２０ｂが実行する処理の一例を説明するフローチャートである。このフローチャートが開始されると、以下の処理が実行される。

［Ｓ２０］リンカ２０ｂは、初期化ライブラリ２２ａをリンクする処理を実行する。

［Ｓ２１］リンカ２０ｂは、他のライブラリ（例えば、数学関数ライブラリ）をリンクする処理を実行する。

［Ｓ２２］リンカ２０ｂは、オブジェクトファイルから新データセクションに係るオブジェクトコードを検索する。

［Ｓ２３］リンカ２０ｂは、新データセクションに係るオブジェクトコードが存在しているか否かを判定し、存在している場合にはステップＳ２４に進み、それ以外の場合にはステップＳ２７に進む。

［Ｓ２４］リンカ２０ｂは、新データセクションの先頭の相対アドレスであるｂｓｓ＿ｓｔａｒｔを取得する。

［Ｓ２５］リンカ２０ｂは、新データセクションの末尾の相対アドレスであるｂｓｓ＿ｅｎｄを取得する。

［Ｓ２６］リンカ２０ｂは、取得した相対アドレスｂｓｓ＿ｓｔａｒｔとｂｓｓ＿ｅｎｄとを初期化ライブラリ２２ａに受け渡す。

【００４２】

具体的には、例えば、これらの相対アドレスを、初期化ライブラリ２２ａの所定の領域に埋め込む。

［Ｓ２７］リンカ２０ｂは、リンク処理を実行する。

【００４３】

次に、図７を参照して、初期化ライブラリ２２ａの処理の一例を説明する。この処理は、ロードモジュール３２が実行され、ＲＡＭ１０ｃ上に配置された場合に実行される処理である。

［Ｓ４０］ＣＰＵ１０ａは、ロードモジュール３２に埋め込まれている相対アドレスｂｓｓ＿ｓｔａｒｔおよびｂｓｓ＿ｅｎｄの値が同一であるか否かを判定し、同一である場合には処理を終了し、それ以外の場合にはステップＳ４１に進む。

【００４４】

ここで、これらの値が同一である場合には、相対アドレスは書き込まれていないデフォルト状態であることを意味するので、その場合には初期化の対象となる動的変数は存在しないとして処理を終了する。

［Ｓ４１］ＣＰＵ１０ａは、例えば、実行時に入力装置１２から供給された初期値を取得する。

【００４５】

なお、この初期値は、後述するように、コンパイル時に指定し、ロードモジュールの所定の領域に格納しておくことも可能である。

［Ｓ４２］ＣＰＵ１０ａは、ｂｓｓ＿ｓｔａｒｔとｂｓｓ＿ｅｎｄによって特定されるＲＡＭ１０ｃ上の領域に対してステップＳ４１で取得した初期値を配置する。

【００４６】

以上の処理によれば、既述した処理を実現できる。なお、以上の例では、初期化の対象となるデータ領域が１つの場合について説明したが、複数の場合には前述の処理を必要回数だけ繰り返すことにより実現可能となる。

【００４７】

次に、本実施の形態の更に具体的な動作について説明する。

先ず、図８を参照して、対象となる変数の新データセクションへの割り当て方法について説明する。

【００４８】

図８の例では、ソースファイル４０の２行目では、配列Ｘ１と配列Ｘ２とが宣言されており、同一のブロック名ＡＬＫが付与されている。また、３行目では、配列Ｘ３が宣言されており、ブロック名ＢＬＫが付与されている。

【００４９】

このようなソースファイル４０がコンパイルされると、オブジェクトファイル４１が生成されることになる。このオブジェクトファイルでは、同一のブロック名が付与された配列Ｘ１と配列Ｘ２とに対しては、同一のデータセクションが割り当てられている。また、配列Ｘ３に対しては、前２者とは異なるデータセクションが割り当てられている。

【００５０】

この例では、ブロック名に応じてデータセクションを割り当てるようにしたが、それ以外にも種々の割り当て方法が考えられる。例えば、配列とそれ以外の変数を分けて別々のデータセクションを割り当てる方法、配列の要素数に応じて割り当てる方法、データの型（４バイト整数、８バイト整数、４バイト実数、８バイト実数）に応じて割り当てる方法、変数名に応じて割り当てる方法、および、手続きの名前に応じて割り当てる方法などがある。

【００５１】

以上の例は、コンパイラが自動的にデータセクションを割り当てる方法の一例であるが、プログラマがコンパイル時にコンパイルオプション等によって直接指定するようにしてもよい。

【００５２】

何れの方法にせよ、データセクションの割り当てを適宜変更することにより、所望の変数のみを所望の値によって初期化することが可能となる。

次に、図９を参照して、ソースファイル内に同一の変数に関する宣言が散在している場合において、これらの変数を統合する処理について説明する。

【００５３】

図９に示すソースファイル５０では、メインプログラムであるＭＡＩＮ内においてブロック名がＢＬＫの配列Ｘ３が宣言されている。また、サブルーチンであるＳＵＢ内において、同一の配列（ブロック名ＢＬＫの配列Ｘ３）が宣言されている。

【００５４】

このようなソースファイル５０をコンパイラ２０ａによってコンパイルすると、ＭＡＩＮ内およびＳＵＢ内のそれぞれに新データセクションに係るオブジェクトコードを有するオブジェクトファイル５１が得られる。

【００５５】

そして、このようなオブジェクトファイル５１に対してリンカ２０ｂによってリンク処理を施すと、オブジェクトファイル５１においてＭＡＩＮおよびＳＵＢ内に別個に存在していた新データセクションに係るオブジェクトコードが統合されて１つにまとめられたリンクモジュール５２が生成される。

【００５６】

従って、ソースファイル内に同一の変数に関する宣言が散在している場合であっても、これらを統合して１つの変数とすることが可能となる。

次に、図１０を参照して、複数のソースファイル内に同一の変数に関する宣言が存在している場合において、これらの変数を統合する処理について説明する。

【００５７】

図１０に示す例において、ソースファイル６０ではブロック名がＢＬＫである配列Ｘ３がＭＡＩＮ内で宣言されており、また、ソースファイル６１では同一の配列Ｘ３がＳＵＢ内で宣言されている。ここで、ソースファイル６０は従来のコンパイラでコンパイルされ、また、ソースファイル６１は本実施の形態のコンパイラ２０ａによってコンパイルされたとする。

【００５８】

その結果、オブジェクトファイル６２では、配列Ｘ３には従来通りのデータセクションが割り当てられ、また、オブジェクトファイル６３では、配列Ｘ３には新データセクションが割り当てられている。

【００５９】

このような２種類のオブジェクトファイル６２，６３を、リンカ２０ｂによってリンクする処理を実行すると、リンカ２０ｂは、オブジェクトファイル６２，６３に含まれている同一の配列Ｘ３に係るデータセクションを比較し、一方が新データセクションで、他方が通常のデータセクションであるので、双方を通常のデータセクションに統一する。

【００６０】

その結果、図１０に示すように、双方の配列Ｘ３が通常のデータセクションに対して割り当てられたロードモジュール６４を得ることになる。

このように、異なるデータセクションが割り当てられた変数をリンク処理によって統合する場合には、通常のデータセクションを優先するようにしたので、同一の変数が異なる領域に割り当てられることを防止し、従来と同様の動作を保証することが可能となる。

【００６１】

なお、以上の実施の形態では、通常のデータセクションに統合するようにしたが、新データセクションに統合するようにしてもよい。その場合には、従来のコンパイラによってコンパイルされたオブジェクトファイル６２の動作が保証できないが、未定義参照の検査は実行することが可能となる。

【００６２】

次に、変数を初期化する際の初期値の設定の方法について説明する。

初期値を設定する方法としては、（１）コンパイル時に指定する方法と、（２）実行時に指定する方法の２通りがある。以下では、先ず、（１）の方法について説明した後、続いて（２）の方法について説明する。

【００６３】

（１）の方法では、コンパイラの起動コマンドに続いて、ソースファイルのファイル名と初期値とを入力することにより初期値を指定する。例えば、図８に示す例のように、ブロック名ＡＬＫとＢＬＫの２種類の新データセクションが存在する場合において、ＡＬＫのみを値“８Ｂ”で初期化する場合の一例を以下に示す。なお、ソースファイルのファイル名は「ａｂｃ．ｆ」であり、コンパイラの

起動コマンドは「ｆｒｔ」であるとする。

【００６４】

ｆｒｔ　ａｂｃ．ｆ　－Ｘ（Ａ）＝８Ｂ

ここで、「－」以降は、コンパイルオプションであり、Ｘ（Ａ）はブロック名の先頭文字がＡである新データセクションに対して初期値“８Ｂ”による初期化を行うことを示している。

【００６５】

以上のようなコマンドによって、コンパイルオプションが指定され、コンパイルがなされると、生成されたオブジェクトファイルの所定の領域には初期値“８Ｂ”が埋め込まれることになる。実行時には、この初期値が読み出されて目的となる変数（この例では配列Ｘ１，Ｘ２）が、指定された初期値“８Ｂ”によって初期化されることになる。

【００６６】

次に、（２）の実行時に初期値を指定する方法について説明する。いま、ロードモジュールのファイル名がａ．ｏｕｔであり、また、図８に示すブロック名がＢＬＫの新データセクションを初期値“８Ｂ”によって初期化する場合には、以下のようにしてロードモジュールを起動する。

【００６７】

ａ．ｏｕｔ　－Ｘ（Ｂ）＝８Ｂ

このようにしてロードモジュールを起動することにより、任意の変数を任意の初期値で初期化してからプログラムを実行することが可能となる。

【００６８】

なお、以上の例では、１６進数“８Ｂ”によって初期化する場合を例に挙げて説明したが、これ以外にも、例えば、非数によって初期化することも可能である。以下は、前述の例において、ブロック名がＢＬＫの新データセクションを非数によって初期化する場合の例である。

【００６９】

ａ．ｏｕｔ　－Ｘ（Ｂ）＝Ｒ４ＮａＮ

ここで、Ｒ４ＮａＮは、４バイトの実数（Ｒ）の非数（ＮａＮ：Not a Number

）によって初期化することを示している。

【００７０】

以上の処理により、任意の変数を任意の初期値によって初期化することが可能となる。なお、コンパイル時において初期値を指定した後、実行時にも初期値を指定した場合には、指定が重複することになるが、そのような場合には何れか一方を優先して採用するようにすればよい。

【００７１】

次に、初期化された変数を用いて未定義参照の検査を行う場合の一例について説明する。

先ず、最も簡単な方法として、ソースコードに対して初期化された変数の内容をプリントアウトする命令を追加し、変数の内容を直接確認する方法について説明する。

【００７２】

図１１は、変数の内容をプリントアウトする命令が追加されたソースファイルの一例を示している。この例では、上から２行目にブロック名がＢＬＫである配列Ｘが定義されており、下から２行目に配列Ｘの第１番目の内容をプリントアウトする命令が追加されている。なお、「ＷＲＩＴＥ」は出力命令であり、括弧内の“６”は出力装置の番号（この例ではプリンタ）を示し、シングルクォートで囲繞された（８Ｚ）は、１６進数で出力することを示している。

【００７３】

このようなプログラムをコンパイル時のオプションとして初期値“８Ｂ”を指定してコンパイルした後に実行するか、または、実行時において初期値“８Ｂ”を指定して実行した場合に、プリントアウトされる値が“８Ｂ８Ｂ８Ｂ８Ｂ”である場合には、未定義で参照されていると判断することができる。

【００７４】

次に、コンパイル時のデバッグオプションを拡張して用いる場合の一例について説明する。

従来、ＦＯＲＴＲＡＮにおいては、コンパイル時にデバッグオプションである「－Ｄｕ」を指定すると、ＣＯＭＭＯＮ変数以外の変数については初期値“８Ｂ

”によって初期化され、この“８Ｂ”が代入された変数が参照された場合には、メッセージ「未定義データが参照されました。」が表示されていた。

【００７５】

ところで、本発明を適用することで、このようなデバッグオプション「－Ｄｕ」の機能を、ＣＯＭＭＯＮ変数にも拡張することが可能となる。

即ち、デバッグオプション「－Ｄｕ」が指定された場合には、ＣＯＭＭＯＮ変数に関しても初期値“８Ｂ”によって自動的に初期化するとともに、変数が参照された際に“８Ｂ”が検出された場合にはエラーを通知するようにすれば、デバッグオプション「－Ｄｕ」の機能をＣＯＭＭＯＮ変数にも拡張することが可能となる。

【００７６】

次に、ＡＮＳＩ（American National Standards Institute）／ＩＥＥＥ（Institute of Electrical and Electronics Engineers）７５４規格に準拠したトラップを用いる方法について説明する。

【００７７】

従来のＦＯＲＴＲＡＮにおいては、目的の変数を非数で初期化する命令をプログラムにうめこんでおくとともに、実行時はオプション「－ｔｒａｐ」を指定することにより、その変数の値が未定義で参照された場合には、メッセージ「無効演算例外が発生しました。」が表示されていた。本実施の形態によれば、このようなトラップを動的変数であるＣＯＭＭＯＮ変数に対して実行時のオプションの指定のみで行うことが可能となる。

【００７８】

以下では、図１２に示すように、ブロック名がＢＬＫの配列Ｘが宣言されており、この配列Ｘが未定義のままで、最後から２行目において参照されているソースファイル８０を例に挙げて説明する。

【００７９】

このソースファイル８０のファイル名がａ．ｆであるとすると、先ず、以下のコマンドを実行することにより、ソースファイル８０をコンパイラ２０ａによってコンパイルする。

【００８０】

ｆｒｔ　ａ．ｆ

次に、得られたロードモジュールａ．ｏｕｔを実行する際には、以下のように、配列Ｘを非数で初期化するとともに、トラップオプションを指定する。

【００８１】

ａ．ｏｕｔ　－Ｘ＝Ｒ４ＮａＮ　－ｔｒａｐ

以上のような指定により、配列Ｘが４バイトの非数によって初期化されて実行される。その場合、図１２に示す下から２番目の代入文が実行される際には、配列Ｘの１番目の要素が参照されるが、未定義である場合には非数が代入されていることから、トラップオプションが動作して、既述したエラーメッセージ「無効演算例外が発生しました。」が表示されることになる。

【００８２】

次に、配列の内容を参照する際に、宣言された範囲を超えた添字が使用されているか否かを判定する場合について説明する。

例えば、図１３に示すソースファイル９０があったとする。このソースファイル９０では、２行目において単精度実数型であって要素数が１０００の配列Ａと配列Ｂとが宣言されている。

【００８３】

このような配列がコンパイルされた場合、従来においては、以下のようなオブジェクトコードが生成されていた。

．ｂｓｓ　ｍａｉｎ．ｌｏｃａｌ，８０００

しかしながら、本実施の形態のコンパイラでは、以下のようなオブジェクトコードが生成される。

【００８４】

．ｂｓｓ　ｍａｉｎ．ｌｏｃａｌ，８００８

即ち、必要な領域よりも８バイトだけ多い領域を確保するオブジェクトコードが生成される。ここで、８００８バイトの内訳を図１４に示す。この図に示すように、８００８バイトの最初の４０００バイトは、配列Ａに対して割り当てられ、続く４バイトは冗長部とされ、続く４０００バイトは配列Ｂに対して割り当て

られ、最後の４バイトは冗長部とされる。

【００８５】

このように、本実施の形態では、配列が宣言された場合には、必要な領域とともに一定の余分な領域が確保され、その余分な領域が冗長部とされる。

このようにして確保した冗長部は、宣言された範囲を超えた添字が使用されたか否かをチェックする場合に使用することができる。即ち、このようにして確保された全ての領域を、例えば、非数によって初期化しておき、実行時にトラップオプションを指定すれば、配列が適切に初期化されている場合には、冗長部のみに非数が格納されていることになるので、メッセージ「無効演算例外が発生しました。」が表示された場合には、宣言された範囲を超えた添字が使用された可能性が高い。例えば、図１３に示す例では、２番目のＤＯループ（６行目～８行目）において、配列添字が１～１００１の範囲で使用されており、添字が１００１となった場合には前述のエラーが出力されることになる。

【００８６】

なお、このようなメッセージだけでは、配列添字が宣言された範囲を超えて使用されたか、それ以外のエラーによるのかは明確ではないが、配列添字のミスの可能性もあることは伺い知ることが可能となる。従来においては、配列添字が宣言された範囲を超えて使用した場合には、何の表示もなされなかったので、エラーの存在に気付くことすら困難であったが、このような実施の形態によれば、エラーが含まれている可能性に気付くことが可能となる。

【００８７】

なお、以上の実施の形態と、既述した方法とを組み合わせれば、動的変数である配列についても、配列添字の検査を行うことが可能となる。

また、このような実施の形態では、従来のデバッグオプションを活用するようにしたので、新たなプログラムを作成する手間を省略するとともに、新たなオプションを付加することによりその処理の分だけ全体の処理速度が低下することを防止することが可能となる。

【００８８】

なお、以上の実施の形態では、主にＦＯＲＴＲＡＮを例に挙げて説明を行った

が、本発明はその他の言語にも適用可能であり、例えば、Ｃ言語その他の言語に対しても適用することが可能である。

【００８９】

また、動的変数を初期化する既述の方法においては、動的変数のメモリ上への格納先を、相対アドレスとして初期化ライブラリに引き渡すようにしたが、本発明はこのような場合のみに限定されるものではない。例えば、ロードモジュールが実行された際には、動的変数が格納されているメモリ上の領域の前後に所定の識別符号を配置するようにし、この識別符号で囲繞された領域を初期化ライブラリによって検出し、指定された初期値で初期化するようにしてもよい。要は、初期化ライブラリに対して、動的変数が格納されているメモリ上の領域を通知するようにすればよい。

【００９０】

最後に、上記の処理機能は、コンピュータによって実現することができる。その場合、情報処理装置が有すべき機能の処理内容は、コンピュータで読み取り可能な記録媒体に記録されたプログラムに記述されており、このプログラムをコンピュータで実行することにより、上記処理がコンピュータで実現される。コンピュータで読み取り可能な記録媒体としては、磁気記録装置や半導体メモリ等がある。市場へ流通させる場合には、ＣＤ－ＲＯＭ(Compact Disk Read Only Memory)やフロッピーディスク等の可搬型記録媒体にプログラムを格納して流通させたり、ネットワークを介して接続されたコンピュータの記憶装置に格納しておき、ネットワークを通じて他のコンピュータに転送することもできる。コンピュータで実行する際には、コンピュータ内のハードディスク装置等にプログラムを格納しておき、メインメモリにロードして実行する。

【００９１】

【発明の効果】

以上説明したように本発明では、動的変数を含むソースファイルをコンパイル処理によってオブジェクトファイルに翻訳し、リンク処理によって実行可能形式のロードモジュールに変換する情報処理装置において、ソースファイルから対象となる動的変数を特定する動的変数特定手段と、動的変数特定手段によって特定

された動的変数が、ロードモジュールの実行時にメモリ上に展開される際に確保される領域を特定する領域特定手段と、領域特定手段によって特定された領域を、所定の初期値によって初期化する初期化手段と、を有するようにしたので、動的変数の未定義参照の検査を行うことが可能となる。

【００９２】

また、本発明では、配列を含むソースファイルをコンパイル処理によってオブジェクトファイルに翻訳し、リンク処理によって実行可能形式のロードモジュールに変換する情報処理装置において、ソースファイルから対象となる配列を特定する配列特定手段と、配列特定手段によって特定された配列において宣言されている領域よりも所定のバイト数だけ多い領域を、ロードモジュールの実行時においてメモリ上に確保する領域確保手段と、領域確保手段によって確保された領域を所定の初期値で初期化する初期化手段と、を有するようにしたので、配列において、宣言された範囲を超えた添字が使用されたことを検出することが可能となる。

【図面の簡単な説明】

【図１】

本発明の動作原理を説明する原理図である。

【図２】

本発明の実施の形態の構成例を示すブロック図である。

【図３】

ＯＳ、コンパイラ、リンカ、ソースファイル、ライブラリ、および、オブジェクトファイルの関係を示す図である。

【図４】

本発明の実施の形態の動作の概要を説明する図である。

【図５】

図３に示すコンパイラにおいて実行される処理の一例を説明するフローチャートである。

【図６】

図３に示すリンカにおいて実行される処理の一例を説明するフローチャートで

ある。

【図７】

図４に示す初期化ライブラリによって実行される処理の一例を説明するフローチャートである。

【図８】

対象となる変数の新データセクションへの割り当て方法を説明する図である。

【図９】

ソースファイル内に同一の変数に係る宣言が散在している場合において、これらの変数を統合する処理を説明する図である。

【図１０】

複数のソースファイル内に同一の変数に関する宣言が存在している場合において、これらの変数を統合する処理を説明する図である。

【図１１】

未定義参照の検査を行う場合のソースファイルの一例である。

【図１２】

ＡＮＳＩ／ＩＥＥＥ　７５４規格準拠のトラップを使用して、未定義参照の検査を行う場合のソースファイルの一例である。

【図１３】

定義された範囲を超えた配列添字の使用を検査するためのソースファイルの一例である。

【図１４】

本実施の形態において確保される配列の領域の一例を示す図である。

【符号の説明】

１　ソースコード

２　動的変数特定手段

３　領域特定手段

４　初期化手段

５　メモリ

１０　情報処理装置

１０ａ　ＣＰＵ

１０ｂ　ＲＯＭ

１０ｃ　ＲＡＭ

１０ｄ　ＨＤＤ

１０ｅ　ＧＣ

１０ｆ　Ｉ／Ｆ

１１　表示装置

１２　入力装置

２０　ＯＳ

２０ａ　コンパイラ

２０ｂ　リンカ

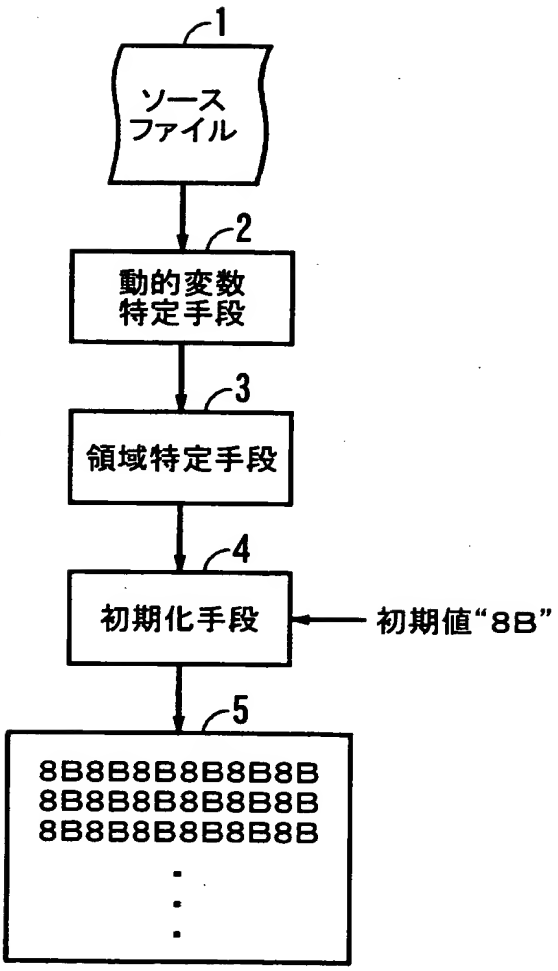
２１　ソースファイル

２２　ライブラリ

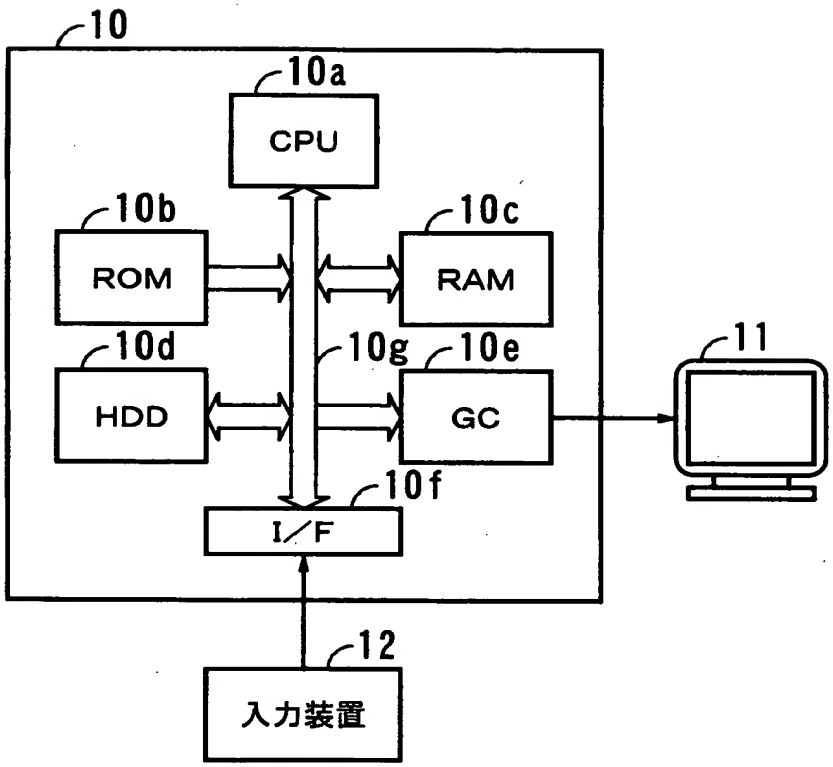
２３　オブジェクトファイル

【書類名】　　図面

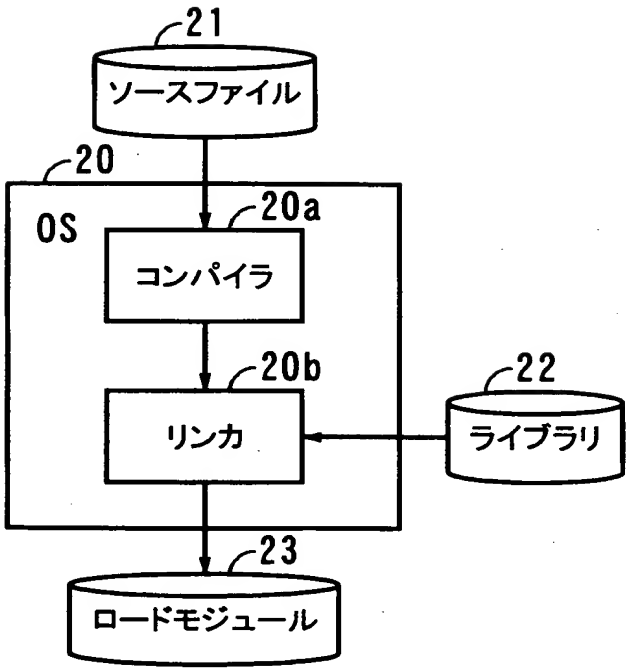
【図１】


1
ソース
ファイル
2
動的変数
特定手段
3
領域特定手段
4
初期化手段
初期値“8B”
5
8B8B8B8B8B8B
8B8B8B8B8B8B
8B8B8B8B8B8B

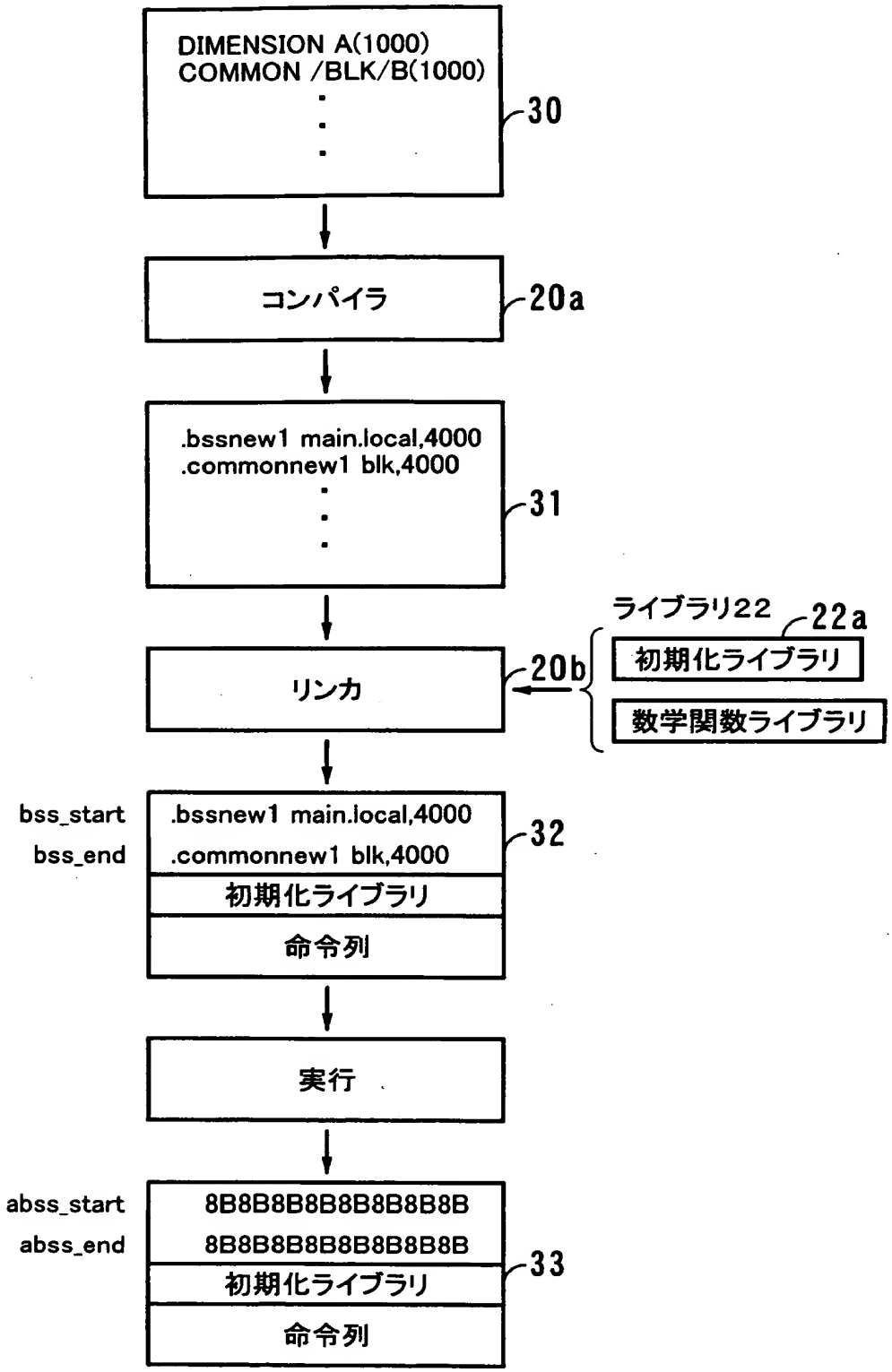
【図２】



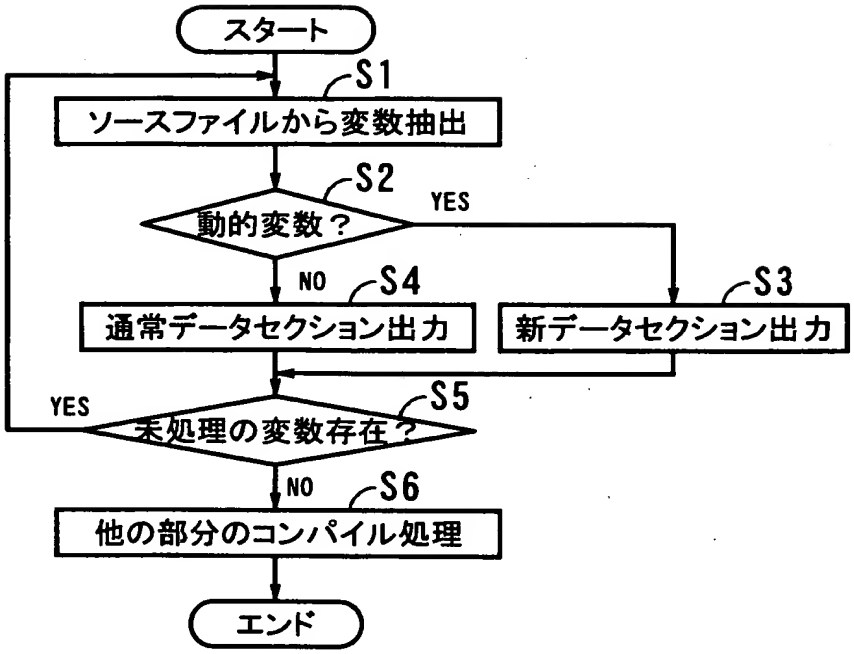
【図３】



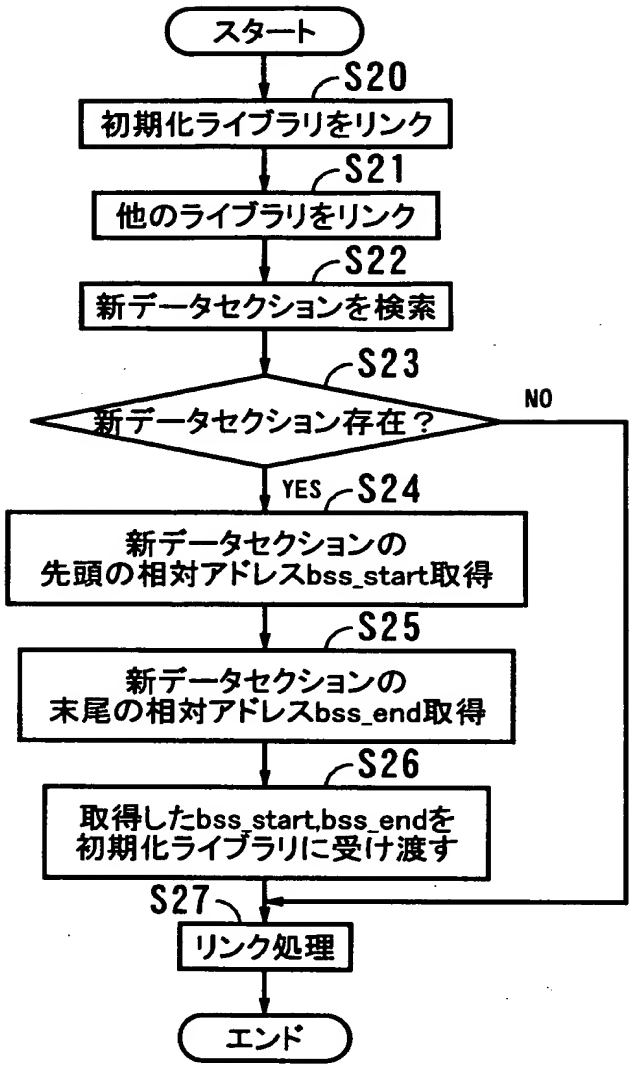
【図４】


DIMENSION A(1000)
COMMON /BLK/B(1000)
30
コンパイラ
20a
.bssnew1 main.local,4000
.commonnew1 blk,4000
31
ライブラリ22
22a
初期化ライブラリ
リンカ
20b
数学関数ライブラリ
bss_start
bss_end
.bssnew1 main.local,4000
.commonnew1 blk,4000
32
初期化ライブラリ
命令列
実行
abss_start
abss_end
8B8B8B8B8B8B8B8B
8B8B8B8B8B8B8B8B
33
初期化ライブラリ
命令列

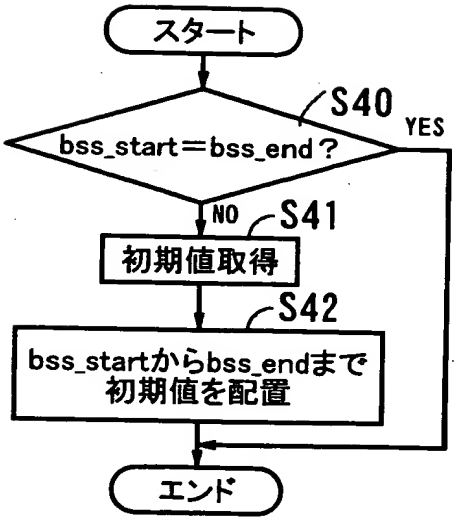
【図５】


スタート
S1
ソースファイルから変数抽出
S2
動的変数？
YES
NO
S4
通常データセクション出力
S3
新データセクション出力
S5
YES
未処理の変数存在？
NO
S6
他の部分のコンパイル処理
エンド

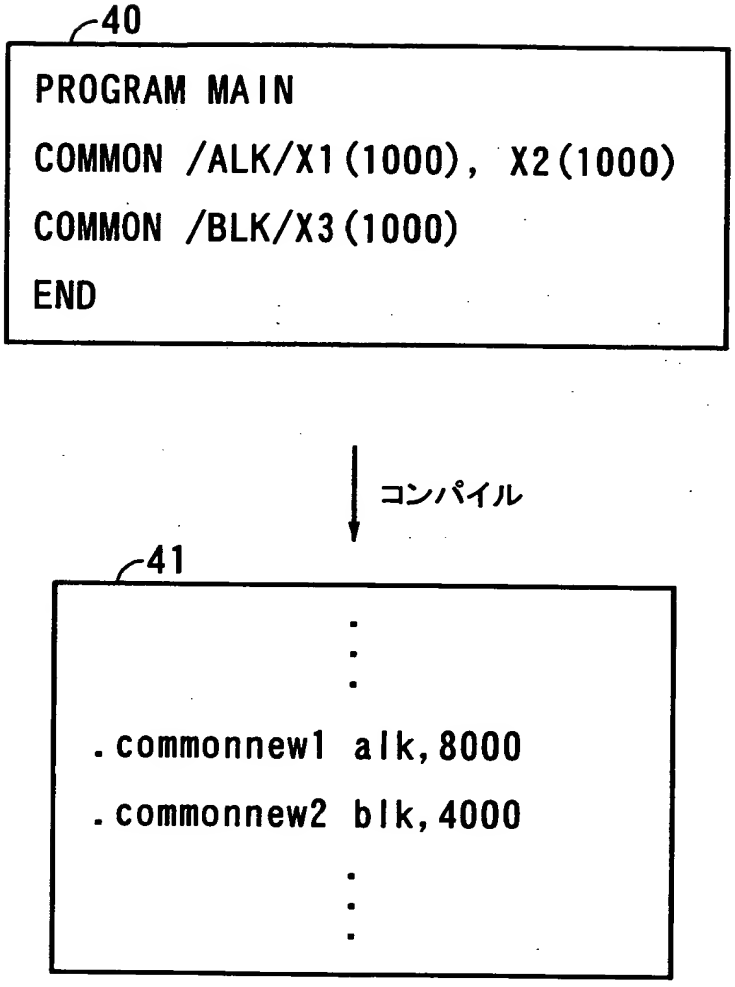
【図６】


スタート
S20
初期化ライブラリをリンク
S21
他のライブラリをリンク
S22
新データセクションを検索
S23
新データセクション存在？
NO
YES
S24
新データセクションの
先頭の相対アドレスbss_start取得
S25
新データセクションの
末尾の相対アドレスbss_end取得
S26
取得したbss_start,bss_endを
初期化ライブラリに受け渡す
S27
リンク処理
エンド

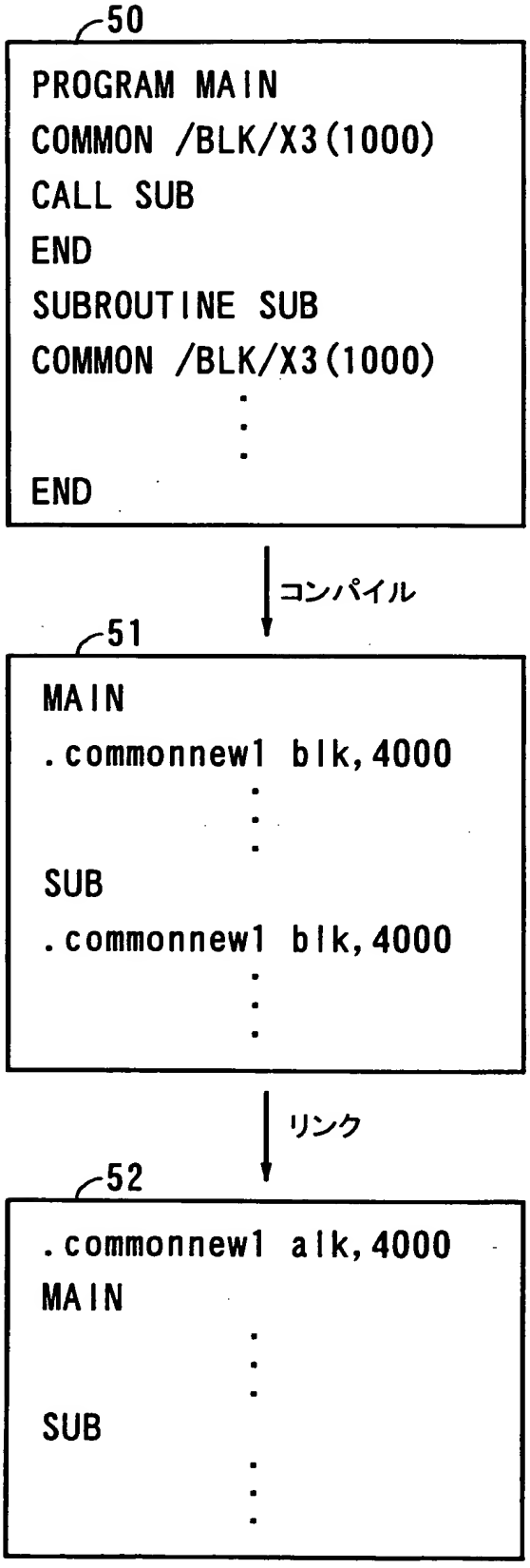
【図７】


スタート
S40
YES
bss_start＝bss_end？
NO
S41
初期値取得
S42
bss_startからbss_endまで
初期値を配置
エンド

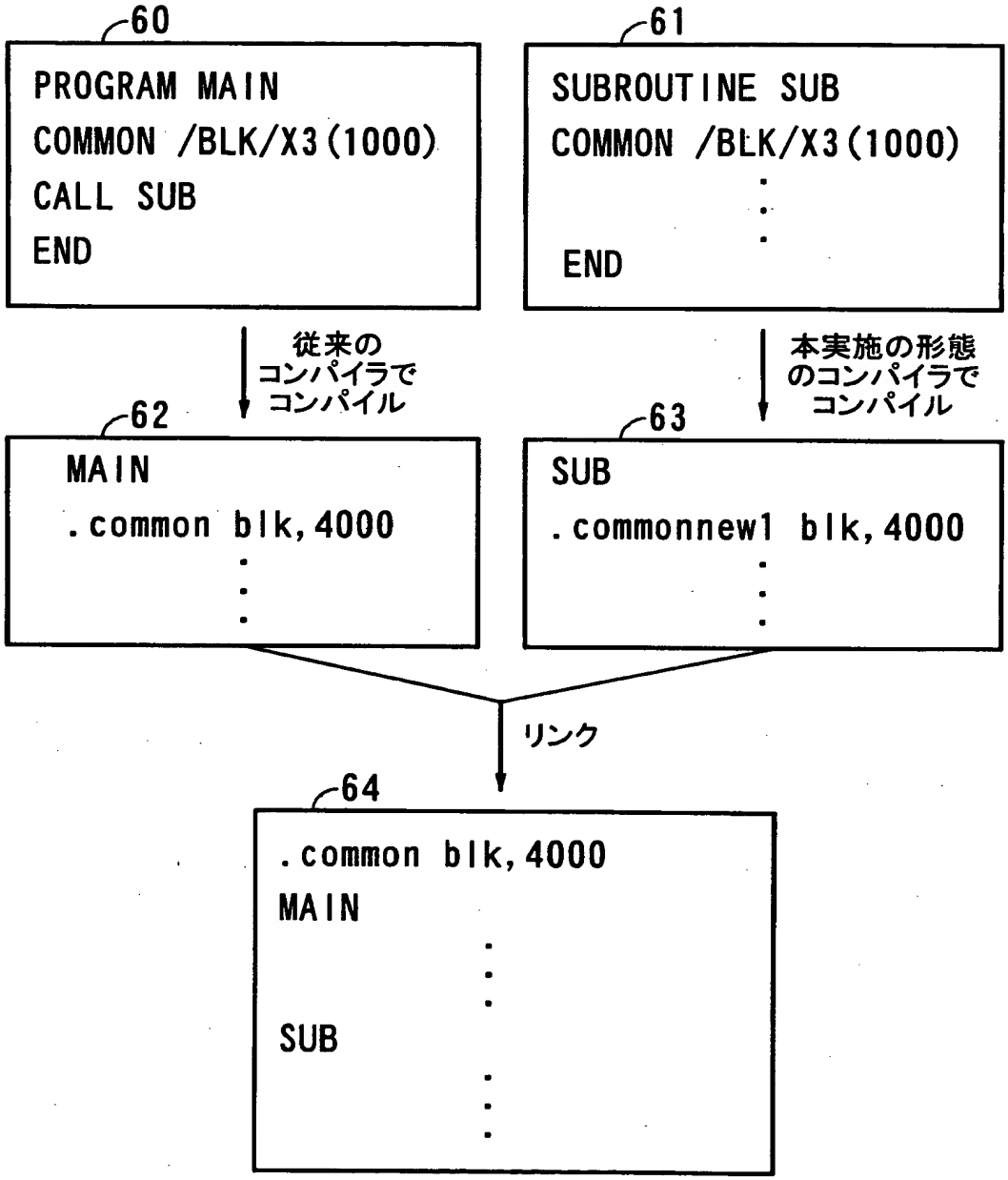
【図８】


40
PROGRAM MAIN
COMMON /ALK/X1(1000), X2(1000)
COMMON /BLK/X3(1000)
END
コンパイル
41
.commonnew1 alk,8000
.commonnew2 blk,4000

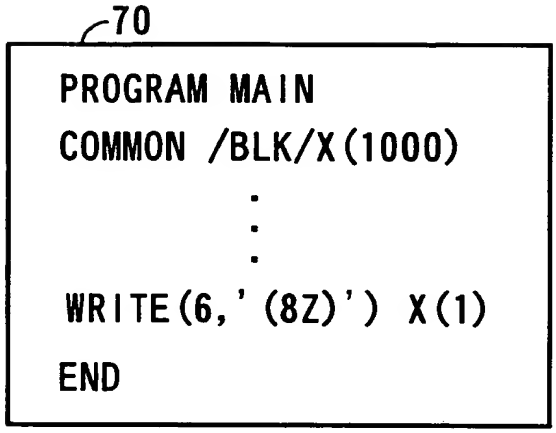
【図９】


50
PROGRAM MAIN
COMMON /BLK/X3(1000)
CALL SUB
END
SUBROUTINE SUB
COMMON /BLK/X3(1000)
END
コンパイル
51
MAIN
.commonnew1 blk,4000
SUB
.commonnew1 blk,4000
リンク
52
.commonnew1 alk,4000
MAIN
SUB

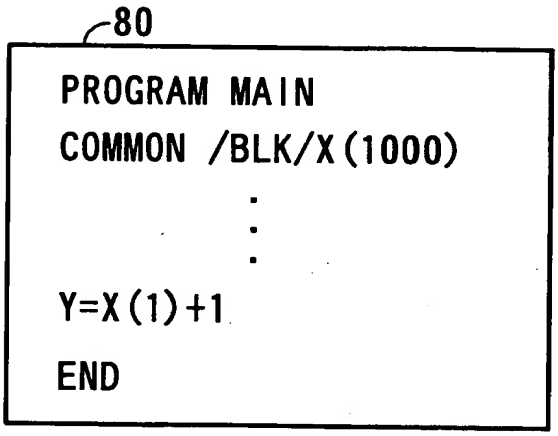
【図１０】


60
PROGRAM MAIN
COMMON /BLK/X3(1000)
CALL SUB
END
61
SUBROUTINE SUB
COMMON /BLK/X3(1000)
END
従来の
コンパイラで
コンパイル
本実施の形態
のコンパイラで
コンパイル
62
MAIN
.common blk,4000
63
SUB
.commonnew1 blk,4000
リンク
64
.common blk,4000
MAIN
SUB

【図１１】


70
PROGRAM MAIN
COMMON /BLK/X(1000)
WRITE(6,'(8Z)') X(1)
END

【図１２】

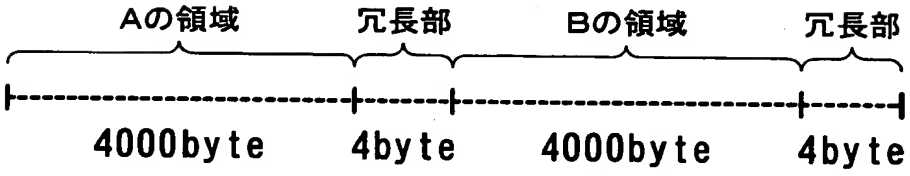

80
PROGRAM MAIN
COMMON /BLK/X(1000)
Y=X(1)+1
END

【図１３】

90

```
PROGRAM MAIN
REAL A(1000),B(1000)
DO I=1,1000
    A(I)=1.0
ENDDO
DO I=1,1001
    B(I)=A(I)
ENDDO
```

【図１４】



【書類名】　　要約書

【要約】

【課題】　動的変数の未定義参照の検査を可能とする。

【解決手段】　動的変数特定手段２は、ソースファイル１から対象となる動的変数を特定する。領域特定手段３は、動的変数特定手段２によって特定された動的変数が、ロードモジュールの実行時にメモリ５上に展開される際に確保される領域を特定する。初期化手段４は、領域特定手段３によって特定された領域を、所定の初期値によって初期化する。

【選択図】　　図１

出　願　人　履　歴　情　報

識別番号　　〔０００００５２２３〕

１．変更年月日　　１９９６年　３月２６日
［変更理由］　住所変更
住　所　　神奈川県川崎市中原区上小田中４丁目１番１号
氏　名　　富士通株式会社